--Many current implementations of Java use the classic mark-sweep-compact method of garbage collection as delivered in the base SUN JVM. References to the objects that are being processed at any instant by the system are stored in the registers, one or more thread stacks and some global variables. The totality of objects that may be needed by the system can be found by tracing through the objects directly referenced in the registers, stacks, and global variables and then tracing through these "root" objects for further references. The objects in use by a system thereby form a graph rooted at the root objects and any extraneous objects are not part of this graph. Once all the objects in the graph are found, the remaining objects in the heaps may be discarded (garbage collected).--

Please replace the paragraph beginning at page 8, line 15 of the specification with the following:

--Referring to Figure 1, there is shown a computer platform 10 such as a Pentium processor based PC typically comprising a motherboard with processor, a 32Mbytes RAM memory, and interface circuitry; a 4G Byte hard drive; a 20x CD ROM; an SVGA monitor; a keyboard and a mouse. On power-on the platform loads into the memory an operating system 12 such as Windows NT v4 and subsequently a Java Virtual Machine (JVM) based on the Java Development Kit (JDK) v1.1 or v1.2. The Java virtual machine is capable of loading a Java application 16 from disk into memory so that the application may be executed on the platform 10. The Java application 16 is object orientated program code and creates Java language objects 18A, 18B, 18C, and 18D. An example of how the objects are built by the JVM is described with reference to Figure 3.--

Please replace the paragraph beginning at page 9, line 8 of the specification with the following:

--The runtime data area 26 comprises thread memory space 28A, 28B, 28n, global heap 34, class area 36 and compiled method area 38. The thread memory space 28 ideally stores local thread objects, local thread execution data and also objects marked global. Local objects are

objects which are used by one thread alone and are not common between. Thread objects are stored in thread heaps 32A, 32B, 32n and thread execution data including object references and register data are stored in the thread stacks 30A, 30B, 30n. The global heap 34 stores objects which are common to more than one thread, that is, global objects. A local object which is loaded into a thread heap 32A for processing by thread 28A may become global if it is later used by another thread, say thread 28B, and may remain in the thread heap 28A or be moved to the global heap 34 by 'stop everything'. Class area 36 stores the classes as they are loaded into the JVM, classes are defined as global because they are common to all the threads hence any field of a class is globally reachable. An object, which is an instance of a class, is placed in a thread heap and is initially local because it may only be used by that thread. An exception is instance objects of class Thread or Runnable which are global at creation because they can be reached by other threads. Whereas an object referenced by a variable in a class is global because the object is reachable from the class which is reachable by all threads. The compiled method area 38 is used by the JIT compiler 24 to store native executable code compiled from the Java byte code.--

Please replace the paragraph beginning at page 13, line 8 of the specification with the following:

--The process of memory management is described with reference to Figure 4. Step 4.1 creates an object from a class stored in the class area 36. Step 4.2 checks to see if the class is a 'global' class, i.e., a class all of whose instances are global or one whose instances we expect to quickly become global, whereby the object is assigned global and/or placed in the global heap (Step 4.7). Step 4.3 calculates the size of the object to see whether it will fit in the thread heap. Step 4.4 performs garbage collection to free up memory if there is not enough space [then garbage collection is performed to free up memory]. Step 4.5 places the object in the thread heap memory along with the object length in multiples of eight. Step 4.6 uses a spare bit in the length attribute as a flag and set as local ('0'). The process ends at step 4.8.--

Please replace the paragraph beginning at page 13, line 22 of the specification with the following: